Trace-Based Decision Making in Interactive Application: Case of Tamagotchi systems

Hoang Nam Ho, Mourad Rabah, Samuel Nowakowski, Pascal Estraillier

Trace-Based Decision Making in Interactive Application: Case of Tamagotchi systems

Hoang Nam HO[1], Mourad RABAH[1], Samuel NOWAKOWSKI[2], Pascal ESTRAILLIER[1]
(1) L3i Laboratory – University of La Rochelle – La Rochelle – France
(2) University of Lorraine – LORIA, UMR 7503 – Nancy - France
{hoang_nam.ho, mourad.rabah, pascal.estraillier}@univ-lr.fr, samuel.nowakowski@loria.fr

***Abstract*— We present our exploratory work for situation preselecting in interactive applications, assuming that the application is an Interactive Adaptive System based on a sequence of contextualized "situations". Each situation confines activities and interactions related to a common context, resources and system actors. When one situation is completed, the system has to determine which is the best following one. We introduce in this paper a new preselecting method that identifies possible next situations among all available situations. We propose a strategy using Naïve Bayes based on the analysis of the sets of available traces (the past of users). Combining all obtained results, we get a set of situations, called set of alternatives that can be used in any decision algorithm. We demonstrate our approach on a case study based on Tamagotchi game.**

Keywords*— *interactive adaptive system, preselection, situations, traces, Bayesian probability, Multi-Criteria Decision Making.

I. INTRODUCTION

Our work considers adaptation in Interactive Adaptive System (IAS) [1]. This kind of systems can adapt their execution according to users behaviours. To control application's scenario structure and unfolding, we confine the interactions in IASs using contextualized blocks called "situations" [2]. A "situation" is one component of the system where actors interact using local resources in a specific context to achieve one or more common objectives. The computation of the application consists in choosing, related to one given situation, the most appropriate following one. This choice is based on a Multi-Criteria Decision Making [3]. The criteria are defined to take into account both local and global objectives of the application. Thus, to choose the next situation to execute, a decision process has been introduced in our previous work [4]. The process starts by identifying the list of the possible situations for the next decision. This shortlisting is not computed by the decision algorithms [3], [5]. In our situation-based context, if we apply these methods to perform directly the decision algorithm on all the available situations, some situations do not need the decision because they are obviously not compatible at a given moment. If we still perform the decision technique on it, computation time will be too high because of the complexity of the decision algorithm. To deal with this problem, we propose to reduce the number of situations that the decision method will analyse. We identify among all the available situations, those that can be executed according to the current state. The current state is a set of properties that contribute to the execution of the application. Then, our problem is: among all available situations, how to preselect a set of possible situations for decision making in IASs.

Concerning our problem, there are some existing techniques that can be used. In [6], [7] authors calculate the distance to determine which item can be recommended for the user. The weaker the similarity of the current state and the previous states gets, the larger the distance is. However, Distance approach drawback is the computing time. In this kind of approaches, we must consider all the data when we want to compute the distance between the new data and the available data. It does not support a model that allows us to avoid to re-compute just some distances with the new data. In another approach [8], authors use the Linear Logic to identify all possible situations; Linear Logic is based on the inputs of the current situation (called pre-condition). This method is very intuitive because it performs only the verification of the logic between the pre-conditions (that are part of situation's structure) and the current state of the system. This method is not flexible when we do not have access to the situation's structure. Moreover, the Linear Logic does not provide a quantified comparison value (as a distance in the Distance approach above). This indicator is necessary to classify the possible situations likelihood. Naïve Bayes is another approach used for recommendation [9], [10]. This method can overcome the mentioned drawbacks by constructing a learning model. The Naïve Bayes aims to compute the executable probability for each situation; this executable probability is then used to classify the available situations. In our context, and knowing its strengths, we decided to choose the Naïve Bayes to perform the preselection computations in an interactive application structured with "situations".

In this paper, we consider an interactive application with many interactions between the actors of the system. These actions can generate information, we call traces [11]. Traces contain many valuable information about the users' past habits and skills; if we use traces, they can help us to deal with the problem more easily, as in [12]. In the present paper, we propose a strategy to use traces for alternatives preselecting.

IEEE Page 1 01/12/y The paper is organized as follows: the section II defines the trace-based that we use; the section III presents a model for situations preselection in situation-based IASs. The case study based on Tamagotchi game is presented in section IV to validate our approach and compare it to others approaches mentioned above. Finally section V concludes the paper with perspectives for further works.

Accepted in IEEE International Conference on Control, Decision and Information Technologies, Metz, November 3-5 2014

II. TRACE-BASED SYSTEM IN INTERACTIVE APPLICATION

There are several works dealing with the Trace-Based System (TBS) [13]. Each one defines a TBS that corresponds to its particular context. In general, all of them are based on the following steps:

- define the set of sensors, which provide the set of traces during the execution of the application;
- collect the primary traces that are the raw traces provided by the defined set of sensors;
- transform the collected traces into modeled ones [14] by formatting and/or filtering according to a given model.

In our context, we consider an interactive application where the user interacts with the application. The execution process is structured as one sequence of situations. At the end of each situation, the system must choose another situation to run according to the current state. In our trace-based system, we define one sensor that labels timestamps, and records what situation is chosen to run every time the decision is made. Our TBS tracks important properties during the execution, and then save them as traces composed of two objects: i) the current state of the application (all of the recorded properties) and ii) the completed situation (situation's identifier). Then we transform these two objects into the following format: the state is a set of properties represented by a vector and the executed situation is transformed into a character string, as shown hereby:

$$trace = pro_1, pro_2, ..., pro_m, situation\_name$$

We now have a simple TBS that supports the data for our proposed preselecting strategy in the next section.

III. TRACE-BASED PRESELECTING STRATEGY IN INTERACTIVE APPLICATION

Fig. 1 describes our trace-based preselecting strategy during the execution of the application.

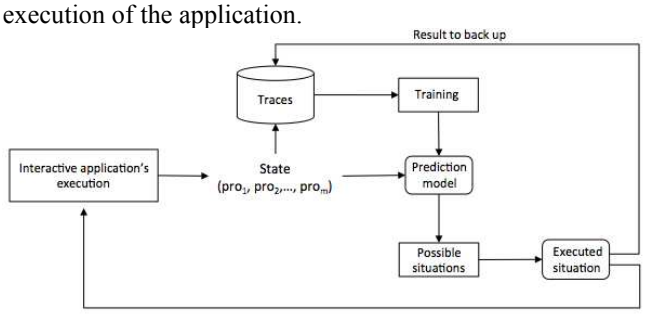

Fig. 1. Process of Trace-Based Preselecting Strategy in Interactive Application

The collected traces are analyzed to predict what are the possible situations according to the current state. Data mining techniques are some of the most efficient methods to solve our prediction problem. Among several existing techniques, Naïve Bayes [15], [16], Neural Network [17], k-Nearest Neighbors [18], Support Vector Machine [19], we decide to use Naïve Bayes. Our choice is motivated by:

- We consider the traces as the primary data for computations. One trace contains heterogeneous information that may be numeric or not. Not all the above techniques can process both numeric and not numeric values, while Naïve Bayes does. For instance, the Neural Network or Support Vector Machine cannot compute with the not numeric data. In fact, the Naïve Bayes is suitable when we add any information type to enlarge the traces.
- Computation time and complexity to analyze the data with Naïve Bayes are less than with others approaches. The Neural Network and the Support Vector Machine require many parameters and the performance of these methods depends strictly on the choice of these parameters. K-Nearest Neighbors is simple and understandable, but it cannot create a training model as other methods.
- Naïve Bayes can create the model faster than others and we do not have to re-estimate the whole model when adding new data.

We will use the Naïve Bayes to preselect a set of situations. We need to predict all the possible situations that can be candidates for next execution step according to the current state. This state (*state*) represents the attributes and properties of the observed system at the end on the current situation execution. For each situation, we compute the probability of its executable ability. We obtain a set of probabilities related to all the situations. A possible candidate situation is the one that has a probability above a defined threshold *h*. If a situation is candidate, we add it to the set of possible situations. The detailed process is presented in the following algorithm. *Alternative(sit_i)* is the preselection status of *sit_i.*

Input: the current state (*state*), set of *n* situations

for each *i = 1* to *n*

Compute *P(sit_i/state)*

if (*P(sit_i/state)* >= *h*)

Alternative(sit_i) = true

else *Alternative(sit_i)* = false

Output: set of possibilities

The Naïve Bayes is based on the Bayes theorem. Given a hypothesis *x* and the object *D*, if we define *P(x/D)* to be the posteriori probability that the object *D* belongs to *x*, this probability is calculated as:

$$P(x/D) = \frac{P(D/x) \times P(x)}{P(D)} \quad (1)$$

where *P(x/D)* is the posterior probability of *x* given *D*, *P(x)* is the prior probability of *x*, *P(D)* is the prior probability of the object *D* and *P(D/x)* is the likelihood which is the probability of *D* given *x*.

The main challenge is to obtain a prediction model *M* that contains all the means μ_k^i and all the standard deviations σ_k^i of the property *k* for the situation *i (sit_i)*. We apply that to the *n*

situations and we will use it to calculate the likelihood of each situation. All the properties that we consider in our context are numeric values and respect the normal (Gauss) distribution, so the likelihood probability is computed as:

$$likelihood\left(pro_k / sit\_i\right) = \frac{1}{\sqrt{2\pi\sigma_k^i}} e^{\frac{-\left(pro_k - \mu_k^i\right)^2}{2\times\left(\sigma_k^i\right)^2}} \quad (2)$$

Using all the likelihoods of each property, we compute the posterior probability of a situation *i* with:

$$posterior\left(sit\_i / state\right) = prior\left(sit\_i\right) * \prod_{k=1}^{m} likelihood\left(pro_k / sit\_i\right) \quad (3)$$

Then, we obtain a set of posteriori probabilities and we compute the probability for each situation with:

$$P\left(sit\_i\right) = \frac{posterior\left(sit\_i\right)}{\sum_{i=1}^{n} posterior\left(sit\_i\right)} \quad (4)$$

This model uses the current state to predict if the considered situation is executable. When we have the updated state vector at the end of the current situation, we use it to predict the execution's ability of all the available situations by computing for each situation its candidate probability. If it exceeds the defined threshold *h*, the situation is considered as a possible situation and all the possible situations constitute the input set of alternatives for the decision algorithm.

IV. CASE STUDY AND DISCUSSION

We need to define an interactive application to demonstrate our approach. The chosen application must be suited to situation-based structuring: the system's execution can be divided into independent sequences performed in a given fixed context. These sequences will correspond to the different system's situations. During the execution, the state of the system will change according to each particular context. The system's execution is hence situations linking all along the execution. To perform the next execution step, the system and/or the user has to choose the next situation to execute in the set of current candidates among all the available situations. The Tamagotchi game suits well to the hypothesis above.

The game describes the life of a virtual pet, named Tamagotchi. The user that plays Tamagotchi should perform various actions that aim to keep the pet alive. We consider his life from the beginning: Tamagotchi was originally an egg and the user must take care of it since its hatching. This game can be structured using situations. The user must successively execute these situations to play the game. When the user comes to the end of a situation, he will obtain an output state of the system and he has to decide among all possible situations the one to execute at the next time. The purpose of this case study is not to offer a completed game, but to have a prototype that will allow us to validate our proposition.

We have identified eight situations that are: *feeding, cleaning, playing, treating, sleeping, socializing, educating and death*. We do not describe in detail all these situations; their names are explicit enough. Once one situation is completed, we must choose the one among 8 situations to continue the game. Then, we define the state of the system using the 6 Tamagotchi's properties: satiety (*sat*), tiredness (*tir*), sadness (*sad*), care (*care*), friendship (*fri*) and politeness (*pol*). If the user wants to play this game, he has to choose, at each step, among the possible situations, which is the suitable one according to the current state of the system.

The TBS in our case study is built, as mentioned in the section II, in three steps:

- Defining sensors: we define two sensors: one to measure the changes in the 6 properties above; the other is responsible for the executed situation.
- Collecting primary traces: we collect the information by the defined sensors.
- Transforming: we extract the updated value of each property, we combine the 6 properties into a vector and we add the chosen situation name at the end of the record.

TABLE I. RECORD'S FORMAT IN TBS

Relation: Tamagotchi

No.	satiety Numeric	tiredness Numeric	sadness Numeric	care Numeric	friendship Numeric	politeness Numeric	**situation** Nominal
1	0.15	-0.13	-0.19	-0.28	3.0	-0.31	treating
2	-0.05	0.28	0.21	0.15	9.0	0.57	feeding
3	0.21	0.02	-0.45	0.05	5.0	0.1	playing
4	-0.0	-0.87	-0.1	-0.73	3.0	-0.88	sleeping
5	0.22	-0.3	0.1	-0.24	7.0	-0.69	sleeping
6	-0.65	0.59	-0.01	0.68	5.0	-0.43	feeding
7	-0.36	-0.36	0.58	0.19	4.0	0.34	sleeping
8	0.34	0.34	-0.06	0.14	2.0	0.24	cleaning
9	-0.45	0.1	0.69	-0.37	5.0	-0.03	feeding
10	0.15	-0.54	-0.1	0.31	8.0	0.75	sleeping
11	-0.09	-0.21	0.6	-0.21	8.0	0.16	sleeping
12	-0.01	0.61	-0.3	0.63	1.0	-0.56	playing
13	0.01	-0.09	0.46	0.4	7.0	-0.18	sleeping
14	-0.58	0.12	0.0	-0.14	1.0	0.32	feeding

Table I gives a sample of traces that we have in the TBS. Each trace has 6 defined properties and the final element is the executed situation, for example: *playing, sleeping, death*… Actually, we are working on the Tamagotchi prototype so we do not have complete real data for the game execution. Therefore, we have created a base of traces[1] to test our method. Statistically, we have created 10020 traces that contain 2273 *feeding* situations, 233 *cleaning* situations, 891 *playing* situations, 2304 *treating* situations, 2269 *sleeping* situations, 1380 *socializing* situations, 534 *educating* situations and 136 *death* situations. We collected the opinion of many users about the game experience in order to correctly simulate what situation is chosen to execute in a specific system's state (combination of the 6 properties' values). We used this information to build a database for our prediction model.

To define the prediction model we started with a training phase. We carried out all the records from the TBS and applied the approach presented in the previous section. Then we have evaluated the performances of the obtained predicting model by using the Weka software [20]. In the Table II, we summarized the correct rate and the needed time to compute

1 "Tamagotchi Traces": https://app.box.com/s/5feoqqmsu39stbmq3l0g

the prediction model using the 4 methods mentioned in Section III above.

TABLE II. COMPARISON OF THE PERFORMANCE OF THE 4 METHODS: NAIVE BAYES, KNN, NEURAL NETWORK, SVM

Methods	Correct Rate	Time for building model (time unit)
Naïve Bayes	83.61%	1 unit
kNN	76.88%	Do not need model
Neural Network	84.03%	14 units
SVM	84.92%	6 units

We can see that the performance of kNN technique is the lowest. And there is no great difference between the three other methods while the computation time for of SVM and Neural Network is longer than for the Naïve Bayes.

We then illustrate how to identify appropriate situation using the obtained prediction model when we have a new state vector. For example, if the observed state vector during the execution of the application is: *state = (sat = 0.03, tir = 0.26, sad = 0.04, care = 0.09, fri = 2, pol = -0.7)*. We want to check what are the situations that can be executed according to this observed state vector. We derive in detail a calculation of the likelihood of the *feeding* situation with $\mu_{sat}^{feeding} = -0.45$ and $\sigma_{sat}^{feeding} = 0.22$ that are respectively the mean and the standard deviation of the satiety property.

The likelihood of the property *sat = 0.03* is:

$$likelihood(sat = 0.03 / feeding) = \frac{1}{\sqrt{2\pi} \times 0.22} e^{\frac{-(0.03+0.45)^2}{2\times 0.05^2}} \approx 0.19$$

Then the likelihood of all properties are computed:

$$likelihood(tir = 0.26 / feeding)$$
$$likelihood(sad = 0.04 / feeding)$$
$$likelihood(care = 0.09 / feeding)$$
$$likelihood(fri = 2 / feeding)$$
$$likelihood(pol = -0.7 / feeding)$$

Finally, the prior probability of *feeding* situation is computed with *prior (feeding) = 2273/10020*.

$$posteriori(feeding / state) = prior(feeding) *$$
$$likelihood(sat = 0.03 / feeding) * likelihood(tir = 0.26 / feeding) *$$
$$likelihood(sad = 0.04 / feeding) * likelihood(care = 0.09 / feeding) *$$
$$likelihood(fri = 2 / feeding) * likelihood(pol = -0.7 / feeding) \approx 0.0012$$

After computing the likelihood of all situations, we apply (3) to calculate the probability of each situation by defining the threshold $h = 15\%$ and obtain the set of alternatives (Table III).

TABLE III. RESULT OF SITUATION PRESELECTING IN TAMAGOTCHI

Situation	Probability	Result
feeding	**21.662%**	alternative
cleaning	0.0003%	non-alternative
playing	11.128%	non-alternative
treating	**20.3%**	alternative
sleeping	2.338%	non-alternative
socializing	**16.863%**	alternative
educating	**27.127%**	alternative
death	0.579%	non-alternative

TABLE IV. THE RESULT OF THE PRESELECTING OF (I) OUR APPROACH VS (II) THE LINEAR LOGIC APPROACH AND (III) THE DISTANCE APPROACH

Situations	State 1 *sat = 0.03 , tir =0.26 sad = 0.04, care = 0.09 fri = 2, pol = -0.7*			State 2 *sat =-0.1,tir = 0.01 sad=0.7,care=-0.1 fri=4, pol=0.1*			State 3 *sat =0.7, tir = -0.2 sad=-0.5, care= 0.2 fri=3, pol=0.3*			State 4 *sat =-0.1,tir = 0.3 sad=-0.2, care=0.5 fri=1, pol=0.3*			State 5 *sat = 0.4, tir = 0.1 sad= 0.5, care= 0.45 fri=4, pol=0.2*			State 6 *sat =-0.5, tir = 1 sad= 0.5, care= -0.1 fri=1, pol=0.9*		
	(i)	(ii)	(iii)	(i)	(ii)	(iii)	(i)	(ii)	(iii)	(i)	(ii)	(iii)	(i)	(ii)	(iii)	(i)	(ii)	(iii)
Feeding	X	X	X	X	X	X				X	X	X				X	X	
Cleaning		X						X	X		X	X	X	X	X			
Playing		X			X		X	X	X		X	X		X			X	
Treating	X		X	X		X												
Sleeping			X			X	X	X	X				X		X			
Socializing	X	X	X		X		X		X	X	X			X	X		X	
Educating	X	X	X												X			
Death																		

We have also evaluated the performance of (i) our approach compared to two existing approaches: (ii) Linear Logic and (iii) Distance approach mentioned in Section I. Table IV summarizes and compares the results of the preselection ability on these approaches. We tested the preselection ability on 6 system's possible states. We see that our method can filter the situations that are alternative for the decision. For each state, we indicate the preselection performance for each approach (an "X" is associated with the situation when the situation is preselected as alternative). Similarities between the three approaches point out that our approach identifies the results as well as the two others, but we can see that our approach is more precise and decrease the number of alternatives for the decision phase. For example, our approach computes the probability of the execution's ability for each situation; the Distance approach computes a distance index is, whereas, the Linear Logic approach does not return a quantified index for each preselected situation as two others. Besides, the Linear Logic needs to verify the current state with the pre-conditions (in the situation's structure). It depends strictly on the states transition. While the Distance method and our approach do not need to consider the predefined structure of the situation; we observe only the current state to compute. However, if we use Distance method, we must verify that the distance cannot

exceed a predefined threshold. But it is also an inconvenient for this method: we cannot preselect any situation if we define a too small distance threshold (for example in the Table IV for the state 6, the Distance approach has no results). Our method must define also the threshold *h* but the performance of the Distance approach is lower than Naïve Bayes according to the Table III.

Although our predicting model is based on simulated data, the results are very promising. It encourages us to complete the Tamagotchi prototype in order to build a real prediction model on real data. We also wonder if we can improve the alternative preselection by combining these three methods. For instance, a suggestion could be to use first the Linear Logic to get a set of executable situations and then to apply our approach to reduce this set. If we combine two methods, we can reduce effectively the number of situations for the decision technique. Besides, our approach has some limitations. It is efficient only if we have enough trace records. During the initial executions, we do not have enough information to compute the prediction model. In this case, users must decide by themselves. Another key issue of our method is the setting of the threshold *h*. The number of alternatives depends strictly on this value.

V. CONCLUSION

In this paper, we have presented a strategy for situations preselection in situation-based interactive systems. Our approach is based on the analysis of the generated traces during the execution process. We have created a Trace Based System adapted to our context. Then we applied a Naïve Bayes technique in order to analyze these traces. Our aim is to build a prediction model that helps us to identify what situation can be executed according to the current state. Our approach doesn't modify the structure of the situations. We only use past states of the system, recorded as system traces.

The main contribution of this paper is the preselection of alternatives for the decision algorithm. We applied it on a Tamagotchi game case study to illustrate our approach and compare it to other existing approaches.

Our future work focuses on new strategies to choose the best situation using the Multi-Criteria Decision Making techniques among the identified alternatives. The defined algorithms will be integrated in a Situation Decision Tool, a situation engine for all interactive applications based on situations.

REFERENCES

[1] P. Brun and M. Beaudouin-Lafon, "A taxonomy and evaluation of formalisms for the specification of interactive systems," in Proceedings of the HCI'95 conference on People and computers, 1995, pp. 197–212.

[2] F. Trillaud, P. T. Pham, M. Rabah, P. Estraillier, and J. Malki, "Situation-Based Scenarios for E-learning," in Proceedings of IADIS e-learning 2012, 2012, pp. 121–128.

[3] M. Köksalan, J. Wallenius, and S. Zionts, Multiple Criteria Decision Making: From Early History to the 21st Century. World Scientific, 2011.

[4] H. N. Ho, M. Rabah, S. Nowakowski, and P. Estraillier, "Trace-Based Weighting Approach for Multiple Criteria Decision Making," Journal of Software., vol. 9, no. 8, pp. 2180–2187, 2014.

[5] S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach. Prentice Hall, 2010.

[6] R. Burke, "Hybrid Web Recommender Systems", in The Adaptive Web: Methods and Strategies of Web Personalization, LNCS: 4321, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Berlin, Heidelberg: Springer-Verlag , 2007, pp. 377–408.

[7] W. Cheetham, "Global Grade Selector: A Recommender System for Supporting the Sale of Plastic Resin," in Proceedings of the 5th International Conference on Case-based Reasoning: Research and Development, 2003, pp. 96–106.

[8] K. Dang, P. Pham, R. Champagnat, and M. Rabah, "Linear Logic Validation and Hierarchical Modeling for Interactive Storytelling Control," in LNCS: 8253, 10th Advances in Computer Entertainment (ACE 2013), Boekelo, The Netherlands, November 2013, pp. 524–527.

[9] K. Miyahara and M. Pazzani, "Improvement of Collaborative Filtering with the Simple Bayesian Classifier," Information Processing Society of Japan Journal, vol. 43, no. 11, 2002.

[10] K. Wang and Y. Tan, "A New Collaborative Filtering Recommendation Approach Based on Naive Bayesian Method," in Proceedings of the Second International Conference on Advances in Swarm Intelligence - Volume Part II, 2011, pp. 218–227.

[11] J. Laflaquière, L. S. Settouti, Y. Prié, and A. Mille, "Trace-Based Framework for Experience Management and Engineering," in Proceedings of 10th International Conference, KES 2006, 2006, pp. 1171–1178.

[12] R. Doumat, E. Egyed-Zsigmond, and J.-M. Pinon, "User Trace-Based Recommendation System for a Digital Archive.," in LNCS: 6176, ICCBR, 2010, pp. 360–374.

[13] L. S Settouti, Y. Prié, D. Cram, P-A. Champin and A. Mille, "A Trace-Based Framework for supporting Digital Object Memories," in Proceedings of 1st International Workshop on Digital Object Memories (DOMe'09) in the 5th International Conference on Intelligent Environments (IE 09) Barcelone , Spain, 2009.

[14] D. Clauzel, K. Sehaba, and Y. Prié, "Enhancing synchronous collaboration by using interactive visualisation of modelled traces," Simulation Modelling Practice and Theory, vol. 19, no. 1, pp. 84–97, 2011.

[15] P. Domingos and M. Pazzani, "On the optimality of the simple Bayesian classifier under zero-one loss," Machine Learning, vol. 29, no. 2–3, pp. 103–130, 1997.

[16] D. J.Hand and K. Yu, "Idiot's Bayes - not so stupid after all?," International Statistical Review, vol. 69, no. 3, 2001.

[17] P.-N. Tan, M. Steinbach, and V. Kumar, Introduction to data mining. Wesley: Pearson Addison, 2006.

[18] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 Algorithms in Data Mining," Knowledge and Information System, vol. 14, no. 1, pp. 1–37, 2007.

[19] V. Vapnik, "The Nature of Statistical Learning Theory," Springer-Verlag, New York, 2000.

[20] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," SIGKDD Exploration, vol. 11, no. 1, pp. 10–18, 2009.